TIME LAPSE HDR: TIME LAPSE PHOTOGRAPHY WITH HIGH DYNAMIC RANGE IMAGES

A Thesis

by

BRIAN CLARK

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2005

Major Subject: Visualization Sciences

TIME LAPSE HDR: TIME LAPSE PHOTOGRAPHY WITH HIGH DYNAMIC RAGE IMAGES

A Thesis

by

BRIAN CLARK

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Ergun Akleman (Chair of Committee)	Karen Hillier (Member)
John Keyser (Member)	Phillip Tabb (Head of Department)

May 2005

Major Subject: Visualization Sciences

ABSTRACT

Time Lapse HDR: Time Lapse Photography With High Dynamic Range Images.

(May 2005)

Brian Clark, B.S., University of Illinois

Chair of Advisory Committee: Dr. Ergun Akleman

In this thesis, I present an approach to a pipeline for time lapse photography using conventional digital images converted to HDR (High Dynamic Range) images (rather than conventional digital or film exposures). Using this method, it is possible to capture a greater level of detail and a different look than one would get from a conventional time lapse image sequence. With HDR images properly tone-mapped for display on standard devices, information in shadows and hot spots is not lost, and certain details are enhanced.

ACKNOWLEDGMENTS

I would like to thank various people who helped and made things easier for me along the way.

My committee members, John Keyser and Karen Hillier, lent their time and expertise. Don House, though not on my committee, was generous enough to do the same. My chair, Ergun Akleman, not only lent his time and expertise but set me on the HDR course and hashed out the initial problem of implementing Fattal's method with me.

Vinod Srinivasan and Robert Knopf offered invaluable problem solving advice. Piotrek Krawcyzk was an excellent and enthusiastic software tester and never tired before I did. Sean O'Malley saved me countless hours with AHDRIA. Felice House lent me her Canon G3, which proved to be an excellent image capture device, and Glen Vigus somehow always found time to provide imaging hardware support. Don Lake provided inspiration for this and other projects through his watercolor magic.

Raanan Fattal was generous in sharing his time and advice with a stranger and even spent valuable SIGGRAPH time talking with me about my project. Greg Ward was responsive, helpful, and down to earth, and Paul Debevec generously shared his huge data sets with me.

Most of all I thank, my wife, Charu, whose presence kept me happy during my graduate studies, and who did her best to kept me focused. I am thankful she arrived at Texas A&M at the same time as I did.

I also wish to thank my wife's cat, Ink, who inspired some analogies by being a great HDR subject. No animals were harmed in the making of this thesis.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

This thesis presents an approach to a pipeline for creation of HDR (High Dynamic Range) time lapse image sequences. A few frames of an example of HDR time lapse photography are shown in Figure 1. In order to understand the different parts needed for creation of HDR time lapse image sequences, it would first be beneficial to discuss the differences between HDR images and LDR (Low Dynamic Range), or conventional images. This chapter covers this topic, and then discusses further the implications of using HDR images and how to display such an image.

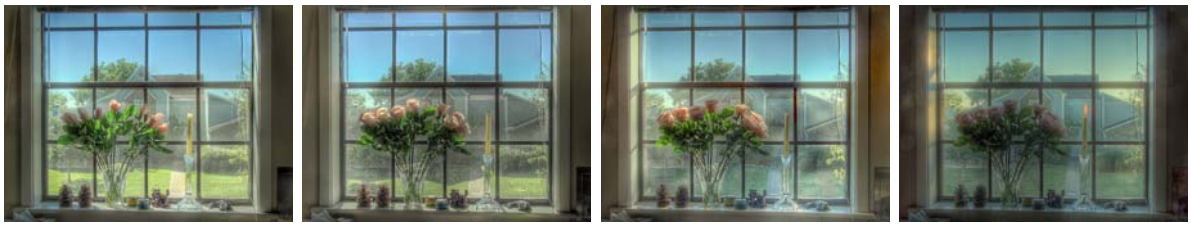


Figure 1. Four frames from the "Flowers" time lapse sequence. The shadows move and the flowers open over the course of the sequence. Such exposures would not be possible with conventional time lapse photography.

1. LDR vs. HDR

Conventional LDR (Low Dynamic Range) digital images have the limitation that a single exposure can capture only a small part of the range of light intensities visible to the human eye at one time. Anyone who has ever taken an indoor picture of someone

---

This thesis follows the style and format of *IEEE Transactions on Visualization and Computer Graphics*.

Figure 2. With conventional photography, the user must choose between exposing for indoor or outdoor objects.

in front of a window on a sunny day has witnessed this effect. The result is that the camera can only expose for the people and the indoor part of the scene, which means that the outdoor scene will be blown out, or the camera exposes for the outdoor part of the scene, which means that the people are underexposed, and therefore too dark (Figure 2).

But with High Dynamic Range (HDR) images, it is now possible for one image to record the range of dark to light that is detectable by the human eye. HDR images can simultaneously record detail in shadowed areas, highlight areas, and obviously, mid-range areas (Figure 3).

HDR images improve over LDR images because there is a severe limitation placed on LDR images in terms of how much information can be stored. A typical conventional LDR image is stored with three channels; R, G, and B (red, green, and blue). Each channel is allocated 8 bits of data per pixel to store the intensity of that particular channel in that particular pixel. 8 bits of data is just enough to hold 256 different integers.

What this all boils down to is the fact that for each color visible on a computer

Figure 3. Tone-mapped version of HDR image constructed from five LDR images shows full range of brightness with details.

monitor, an integer value of 0 to 255 is available to describe the intensity of the pixel for the red, green, and blue component. Though it may seem like a lot of data while mousing around in Photoshop or Gimp, it is actually quite limited when one considers the full range of light bouncing around in the world. For example, consider the following scene: My wife's cat, Ink, is lying on a white sheet of paper in the sun. Ink got her name because of the color of her fur, which is not blue. The clever reader has already deduced that the cat's color is black, but I will now divulge the fact that it is black so that the discussion may move forward. Ink also has white paws and belly, and pink skin, which is visible when the cat is shaved, but the skin color when shaved (and how I know the skin color) is irrelevant for this discussion.

Returning to a digital image of the scene described: Ink is lying on some white paper in the sun. In order to get the most out of the available storage in a conventional RGB image, both a black point and a white point are chosen either by the software, or manually by the user. The black point is the point at which there is determined to be no useful information darker than this, so the RGB value of black, or (0,0,0) is assigned to this, and all things darker. The white point is where there is determined

to be nothing brighter, and is given the RGB value of white, or (255,255,255). All pixels in between are given values in between, which should come as no great surprise, even to those readers lacking in powers of deduction.

While this works for getting the idea across that there is a black cat lying on some white paper, it may not tell the whole story. For example, what the careful observer of this image doesn't notice, nor *could* the careful observer notice, is that Ink has just devoured most of the last piece of European dark chocolate, and will soon be very sick. In fact, when the person who took the picture entered the room, Ink laid down on top of the remaining chocolate in order to hide her crime. The corner of the piece of chocolate is just barely poking out from beneath her leg, and is barely visible to the eye between the black fur and being in the darkest shadows in the scene. In fact, if one were to measure its brightness from the camera's perspective on a scale from 0 to 255 with 0 being completely black, and 255 being the brightest thing in the scene, it would be at 0.49999, which is certainly not completely black, and might be visible to the careful observer's eye.

Unfortunately, the careful observer is not using his eye – not directly, anyway. He is using his eye through the filter of an LDR image like the one in Figure 4. And since LDR means 8 bits per channel, and 8 bits per channel means that the values from 0 to 255 are *integers*, and the chocolate-in-fur-and-shade value of 0.49999 gets rounded to 0, the chocolate is read by the camera as black – pure unadulterated *zero*. This blends in perfectly with all the black fur in shadow around it, which also has a value of zero, so the careful observer actually has zero chance to notice, and the careful observer ends up with cat vomit all over his carpet, which until recently had an RGB value of roughly (200,200,200).

Another thing that even the careful observer would miss is the quality of paper upon which the cat has established as her place of repose. While the camera reads

Figure 4. LDR image of Ink lying on the chocolate.

the sunlit paper as white, or (255,255,255), it raises the question of *what, exactly, is white?* Surely everyone has had the experience of selecting a piece of white paper, only to find that another sheet of white paper actually was brighter, or whiter, which means that the paper which was assumed to be white is actually just an off-white white.

And would it then be possible to find another sheet of paper which is whiter still? And then another? Frankly, it depends upon how much time one has on one's hands. But one need not look just to white paper. "Whiter" than paper would be a light bulb (while lit, of course), the specular reflection from a car, fresh snow on a sunny day, or the sun, itself. In other words, "white" is relative, at least when using it to describe a reference point for the viewer's perception.

It should be noted that surface properties of an object are not the only factors

that cause it to be perceived as brighter or darker in relation to other objects. The amount of light shining on an object is as important as the color of the object. For example, two identical sheets of paper with lights of different intensity shining on them will appear to be different shades. Taking this to the extreme, even the "whitest" paper will be perceived as black to our eye or any HDR image if there is no light hitting the object, since no light from the object reaches the eye or camera. So the "white" or "off-white white" that is observed or recorded in an image is actually the combination of the surface properties of an object and the intensity of light hitting the object.

Getting back to the matter at hand, this white paper upon which Ink is lounging has a value of (255,255,255) in our digital image wherever the sun shines directly upon it. It sure looks for sure like the white that's called pure, but is it really white, or just an off-white white [1]? Again, the careful observer never had the opportunity to notice that the paper was no ordinary paper because the LDR image didn't show him the whole picture.

In fact, this paper was the costly "blue-line" paper upon which, until recently, every Texas A&M thesis student was required to print his or her thesis. This special off-white white paper is recognizable by three basic characteristics: First, it has a thin faint blue line around the outside which denotes the margins. Second, it has a watermark that says "ATM" and boasts of its recycled content (which may or may not also apply to the thesis printed upon the paper). And third, it is an off-white white.

But since a value must be designated as black, or 0, and a value must be designated as white, or 255, much of the difference in intensities between the faint blue lines and the watermark and whitest part of the paper are so subtle that they were lost when encoded into integers, and the rest of the values were "clipped," which is

what happens when a value exceeds the value which is designated as the extreme, or in this case, our white point. Just off screen, for example, is a bottle of water and some shiny metal objects (trust me). There are even some shiny plastic objects of bright colors. All these objects have light intensities that are brighter than the white paper, because the sun is reflecting and refracting right into the camera on these hot spots.

These values are clipped, or would have been clipped, if the camera had included them in the scene, or if there had been a scene, or a camera or paper, or a cat. Actually, there is paper, and there are cameras, and even cats, and one of the cats is, in fact called Ink, and it does, in fact have pink skin underneath its black and occasionally white fur. But I never got around to taking that picture, because I didn't want my carpet soiled, and I already knew it wouldn't look good anyway for the reasons I just described. Then my advisor got involved, and said I should take just such a picture to illustrate my point, and I protested because I wished to preserve the consistency of my carpet's color. We came to an understanding wherein if I would take the aforementioned picture, I would graduate. Thus the resale value of my carpet was sacrificed for education.

It should be noted that LDR photographs are not inherently unable to display the difference between the paper's off-white white, the watermark's off-white white, and the blue lines. Such an image is possible as long as everything else in the image has a similar brightness. The key here is the dynamic range of the image (which the clever reader already suspected, since those words appear in the title). If the black point and the white point are both on the sunlit paper (and though the black point is not at all black to our eye), all 256 integers can go toward describing this very limited dynamic range, and subtle differences can be shown, as in Figure 5. But if the black point is far away from the white point, the 256 values alloted to cover the dynamic

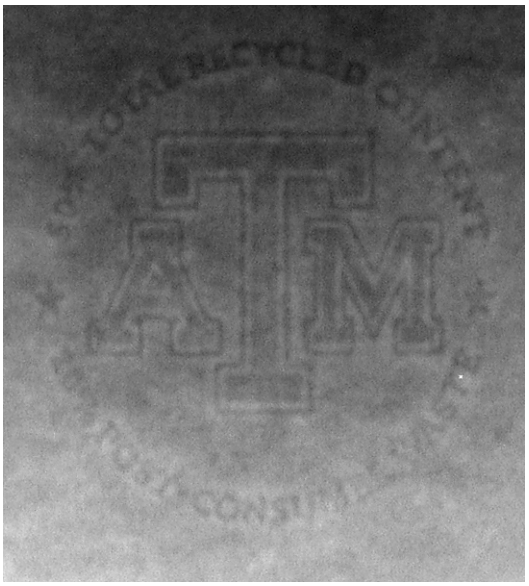


Figure 5. The difference between off-white whites – the watermark. Though there is a very subtle difference between these shades of white, it is possible to display them in a conventional LDR image. All other detail, however, suffers.

range are not sufficient to capture the subtle changes in luminance at the range of intensity of the paper.

2. And HDR Shall Set Us Free...Right?

HDR images are freed from these limitations. First, instead of integers, floating point values are used. While the precision varies depending upon what kind of floating point storage is used and the architecture of the computer, the carpet might have been spared with a high dynamic range image, since 0.49999 would have been read roughly as 0.49999 instead of 0, and the careful observer would have at least had a chance to notice the chocolate underneath Ink's leg (this, of course depends upon how the careful observer views said image, but that's a topic to come shortly), as in Figure 6. Second, HDR images are not bound by the ceiling of 255, so not only could the same image capture the 0.49999 value of the chocolate beneath Ink's leg, it could

also capture details on the paper, and brighter objects, such as the highlights on the various shiny objects sitting next to the paper, and even values up to the light source itself.

It is important to note that HDR images do not have to be stored as floating point numbers, since what is important in the HDR image is the proportion of light intensities. HDR images can be stored using only integers, as long as the ratios remain intact. I have chosen to give my examples using floating point numbers, since they seem intuitive and HDR images are commonly stored using some sort of floating point value.

Regardless of how the data is stored, again, the question comes up: *What is white?* With HDR, the answer is a retort: *Who cares?* Since there is no absolute ceiling, aside from the maximum value of the floating point used, a veritable plethora of off-white whites are possible. The question of what is absolute black is still as easy as before: Zero – unless we want it to be something else. And now negative values are possible. What the negative values signify is up to the user. Perhaps an image could be used as a light intensity map, and negative values could signify where to suppress light. Or in some color models, a negative value is used to express parts of the spectrum [16]. The key here is flexibility. HDR allows everything possible under LDR, and much more.

As wonderful as this is, digital nirvana has not yet been reached. Attaining oneness with the digital and physical universe simultaneously is more complicated than simply encoding data in a new format, since this move raises another issue: *Now that this wonderful image is stored in memory, how can it be viewed?* It could very well have a dynamic range of 100,000:1, but conventional computer monitors have a dynamic range of 100:1 [14]. So not only is the data in a format beyond the capability of the display device, the amount of luminance the device can put out isn't

Figure 6. With a tone-mapped version of the HDR image of Ink, the chocolate is visible just behind her left foot and under her tail.

even close to the dynamic range of the HDR image.

So if it's impossible to display an image of 100,000:1 on a device that is only capable of 100:1, what's the next best thing? The obvious answer: *Fake it.*

Rather, to put it in more scholarly terms: *Give the viewer the illusion that he or she is looking at an image with a high dynamic range, while in fact, it is a low dynamic range image.* But how can this be done? Is it even possible? Given that the topic of this paper relies heavily on it being possible, the clever reader will put his or her money in the "yes" column.

3. Viewing HDR in LDR

In order to create the illusion that this HDR image is being displayed on a standard display device, it must be brought back to that old fashioned 8 bit per channel color space. How can a floating point number with all its precision and a range from 0.001 to 1046.2356 be crammed into integers from 0 to 255? Several simple ways come to mind. Unfortunately, none of them work well.

a. Viewing HDR: Obvious Method #1

The first obvious method with poor results would be to simply interpolate all the values to fit from 0 to 255. But if the image has an object – say a lit light bulb – that's many times brighter than the next brightest object, the light source will be described as 255, and the rest of the image described near 0. This leaves most of the range unused, taken up by the difference between the light intensity and the next brightest object. This yields a very dark and usually grey image, and doesn't adequately express the "Kodak Moment." (Figure 7)

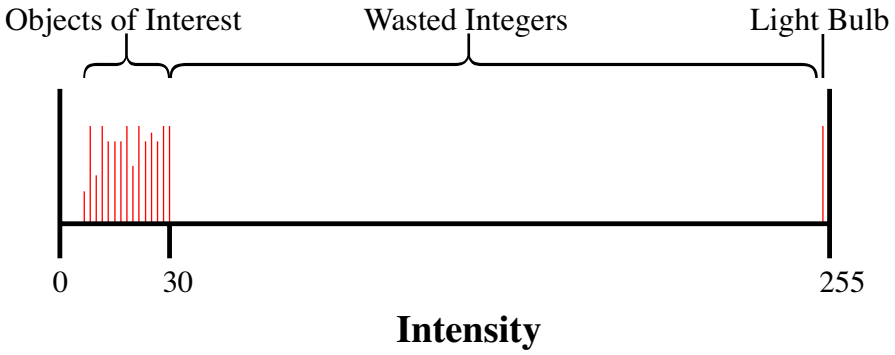


Figure 7. Interpolation leaves the middle of the range from 0 to 255 empty. The area with all the useful information is crammed into the region where we are used to seeing shadows.

b. Viewing HDR: Obvious Method #2

The other obvious method with poor results would be to select a black point and a white point and clip information that falls outside those points. In the above scene with the light bulb, the white point could be set below the light bulb, just above the intensity of the next brightest object. This would knock out the large unused space between the light and the next brightest object, so it wouldn't waste all those precious integers in the middle of the range between 0 and 255. The light and all the details on the light bulb would simply be mapped to 255. The black point, which would be placed just below the brightness of the main object of interest in the scene, would clip everything darker than the object of interest, setting these values to zero. Now the entire range from 0 to 255 is available to describe the main object of interest, and the detail of that object is beautifully displayed.

The downside, of course, is that in order to beautifully display the detail of that object of interest, the detail for the brighter and darker objects was sacrificed.

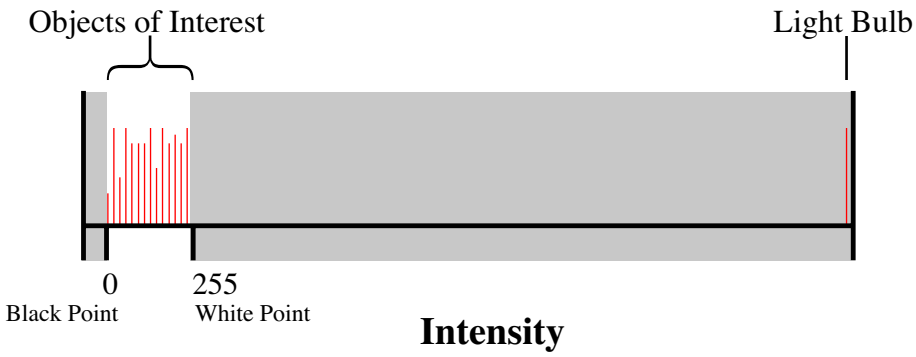


Figure 8. Setting a black point and white point around the object of interest results in clipping large parts of the dynamic range, and therefore lost information in highlights and shadows. The result is a plain LDR image.

Thus, the HDR image was turned back into a LDR image with all its shortcomings intact – shadow detail and highlight detail are lost, and at a lot of cost and trouble (Figure 8). It would have been more efficient to have the digital camera do it directly, since that's what it does anyway – convert high dynamic range light into low dynamic range digital images.

These two obvious solutions are not solutions at all, since nothing is accomplished with these methods. A way is needed to bring this large range of numbers with all its precision into the 0 to 255 integer range, and to do so without destroying the perceived relation between the brightness of the intensities. But is *that* possible?

c. Solving the HDR Display Problem

Fortunately, it is indeed possible, thanks to the gullible human visual system and the efforts of many people who have come before me. In fact, it has been done many many times. The key is to take advantage of the way the human visual system

processes what it sees. It turns out that the human visual system can detect very subtle differences in light intensities over a small localized area, but is not good at judging absolute brightness [15]. This means that it can detect subtle differences in brightness when the areas are directly next to each other, but it cannot tell how bright a scene is in general.

For longer than there have been digital images, computers, or even photography, artists have been exploiting this fact. It is possible to paint a picture that contains fire or some other light source, where the painting is properly "exposed" (to borrow a photography term) to capture detail in the light source, and to also capture detail in shadow on the other side of the picture. The artist pulls off the illusion by varying the brightness of the image locally to express detail, and can use the same values elsewhere in the image to express detail in a darker area, as long as the two areas aren't directly adjacent to each other. An example of this is shown in Figure 9.

Taking a cue from the artists, the computer science community has also been exploiting this characteristic of the human visual system, and calling it tone-mapping. Tone-mapping is simply assigning a value to a pixel to represent an intensity of light. Like a painting, a tone-mapped image does not have to use linear mapping (where pixel brightness is directly proportional to light intensity). With a non-linear tone-mapping algorithm, the same brightness can be used to simultaneously express detail in the brighter area and in the darker area, which means that the exact same integers between 0 and 255 can be used to describe both light and dark areas. Another alternative is to reduce the difference between the brightness of two objects so that the intensities of the pixels are no longer proportional to the difference in real world brightness. Now these 256 integers give us much more mileage than if they were used to map intensity differences proportionally to the way they are in real world scenes.

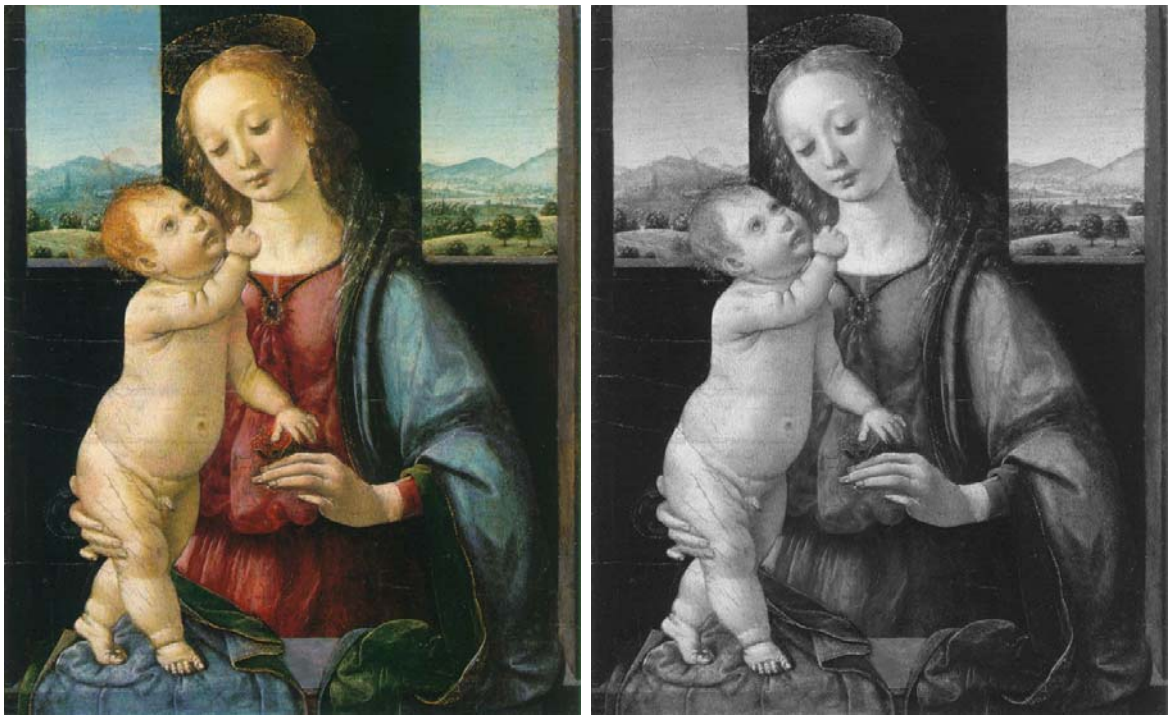


Figure 9. *The Dreyfus Madonna* by Leonardo da Vinci [2]. In this painting and many others, the difference in brightness between the inside and outside doesn't seem unnatural to the eye, but would not be possible with a conventional exposure in natural light. The brightness of outdoor objects is similar to the brightness of indoor objects. A grayscale version has been included on the right to make it easier to see brightness. Reprinted with permission from Mark Harden at http://www.artchive.com.

4. Time Lapse Photography

Time lapse photography offers a view of a scene across time, while conventional photography only provides one slice of time. With time lapse photography, objects that are in transition are apparent to the viewer, so a greater understanding of what is happening in a scene is possible. In this way, it shares a characteristic with HDR photography in that more information is captured than in a conventional photograph.

Another similarity shared by HDR and time lapse photography is the mood that can be brought by each. HDR images have a surreal quality about them, almost as if they were painted or computer generated, since they look like photographs, yet they display the world in a way that it is not normally seen in photographs. The same is true of time-lapse photography. The viewer sees the world in motion, but at a rate that is not normal. Things that normally happen too slowly to be observed are quite apparent in time lapse sequences, and events that normally happen at a comfortable pace for the viewer are suddenly so fast that they are blurred, and pass in an instant.

Combining the surreal qualities of HDR and time lapse photography enhances the surreal qualities of each without adding any perceptual difficulties for the viewer. The viewer has already seen, and is therefore already familiar with time lapse photography. Likewise, the viewer has already seen paintings and illustrations that represent a higher dynamic range of light than one would see in a photograph.

CHAPTER II

HISTORY

In recent years, the use of HDR images has grown, and the number of papers has grown as well. This chapter talks briefly about some tone-mapping algorithms and about some ways to generate HDR images.

1. HDR to LDR Reduction

The computer science community has come up with many different schemes to express HDR images through LDR images, or tone-mapping. In his 1993 paper, "Tone Reproduction for Realistic Images," Jack Tumblin investigated a method for displaying computer generated images that accounted for the difference in human vision and conventional display screens[13]. In 1997, Greg Ward introduced a method for tone-mapping HDR images to LDR images that involved altering the histogram of the image to suppress brighter areas without losing detail in the image[17]. This method produces smooth clean images, and works fairly well, but when looking at an area that is directly sunlit next to an area completely in shade, the image looks as if all the pixels have been divided into bright and dark. This makes the image too "contrasty," as there seems to be no midrange, which is where we are accustomed to viewing the details of an image. Consequently, the image looks as though different areas would benefit from separate histogram adjustments (Figure 10 Top).

Jack Tumblin then released his LCIS, or "Low Curvature Image Simplifier" in his 1999 paper, "LCIS: A Boundary Hierarchy For Detail-Preserving Contrast Reduction" [14]. LCIS looks at the gradient of the intensity and uses a model for heat dissipation (anisotropic diffusion) to determine how to reduce the contrast between

Figure 10. Top: Ward's method doesn't work well across a sharp change in intensity between two areas. Middle: Tumblin's LCIS method has halos and added noise. Bottom: Fattal's method is much cleaner than Tumblin's and works well across a sharp change of intensity between two areas.

large gradients while keeping details. This method works well for bringing everything into a clear visually appealing intensity range, but introduces some noise and halo effect to the image (Figure 10 Middle).

In SIGGRAPH 2002 alone, there were three new methods for such tone-mapping of HDR images, and the results offered in the papers are very impressive. One such method is Durand and Dorsey's "Fast Bilateral Filtering for the Display of High-Dynamic-Range Images." Essentially, this method uses an edge preserving filter and separates an image into a detail layer (high frequency) and a base layer (low frequency), and then reduces the contrast of the base layer to fit into LDR space, while leaving the detail layer unchanged in order to preserve detail[4].

Another method presented in SIGGRAPH 2002 was "Photographic Tone Reproduction for Digital Images" by Erik Reinhard, et al. This method tries to duplicate The Zone System which was developed by Ansel Adams, and is a method used by conventional photographers in order to get a properly exposed photograph even in unusually bright (or "high key") scenes and in dark (or "low key") scenes. This method works by pushing the average intensity, or key of the image into the middle zone. It also looks at areas bounded by high contrast, and uses "dodging" and "burning," two terms borrowed from conventional photography, to preserve detail in bright and dark areas. To do the digital version of dodging and burning, it treats these areas as separate images, choosing a local key for such areas[11].

Fattal and Lischinski submitted "Gradient Domain High Dynamic Range Compression," which looks at the gradient of image intensities and compresses only large gradients, while leaving the small gradients alone. This preserves detail, while bringing the image's dynamic range down to the point where it can be shown on conventional displays[5] (Figure 10 Bottom). I was quite impressed with the results of this method, and the logic behind it appealed to me, so I adopted this as my method.

Further details about this method appear later in this thesis.

2. HDR Image Creation

It is also worth mentioning where HDR images come from to begin with, since alas, they do not grow on proverbial trees (or literal ones, either). One way HDR images can be created is synthetically, by rendering a scene from some imaging software. Radiance is one such package, and is one of the first ones to make use of the High Dynamic Range format. In fact, the native format for Radiance is .hdr, which is still one of the most prevalent formats today.

Another way to create HDR images is to capture HDR data directly using very expensive hardware. Such a camera is generally used to capture HDR environment maps. A company named Spheron makes "SpheroCam HDR," which is capable of capturing 26 f-stops of dynamic range in a single exposure. Unfortunately, this camera comes with a $65,000 price tag, which makes it less than practical for anyone not using this camera on a regular basis in the employ of a studio.

We now come to the last way, which happens to be the most common among plebeians like myself without the means to purchase the aforementioned spiffy hardware. This is to set up a camera on a tripod and take multiple exposures of the same scene, varying the shutter speed for each exposure. The sequence of photographs can then be assembled into a single HDR image by using the data from each of the individual images, combined with the knowledge of the relative exposures of each image. The intensity of each pixel can be calculated by looking at the same pixel in each different exposure[3]. Such a method was presented by Paul Debevec at SIGGRAPH in 1997. Debevec also created software called HDR Shop specifically for assembling HDR images. Greg Ward, creator of the Radiance software package, released soft-

ware called Photosphere that also assembles HDR images from multiple exposures. He also released a command line utility called hdrgen, which I found to be the most useful for time lapse HDR purposes, since command line interface allows for scripting automation. Further description of their use follows later in this paper.

CHAPTER III

METHODOLOGY

This approach consists of four stages: image capture, HDR generation, tone-mapping, and display. The basic process is the same for still images and time lapse sequences. The major difference is that for a time lapse sequence, batches of images must be handled, and the display is through some movie format, such as Quicktime, rather than just a tif or jpg. Of course, the subject matter is a bit different when dealing with time lapse than with still images, and it takes much longer to collect the data.

1. Image Capture

When using the multiple exposure method of capturing data for a single HDR image, it is necessary to take multiple exposures, altering the exposure for the different levels of light. The shutter speed should be varied, rather than the f-stop, since varying the f-stop will introduce differences in focus due to a changing depth of field. The best levels of detail and clarity are attained by starting with a very fast shutter speed – just enough to pick up the brightest highlights in the scene, with none of the pixels reaching a value of 255. For the next exposure, the time that the shutter is left open should be doubled, repeating this until the image is mostly white, with just the darkest objects showing some detail (Figure 11).

It is not, however, absolutely necessary to take an exposure at every f-stop. It is possible to increase the amount of light being let in by two stops (quadruple the amount of light), or even four stops. Some detail and smoothness of the image is sacrificed with a greater distance between stops (capturing four stops apart, for example), but the time to capture and disk space are both greatly reduced. Depending

Figure 11. The small images are one stop apart, and are combined to form the HDR image used to create the large image. The large image is a tone-mapped version of the HDR image.

upon the setup used, the time to capture isn't just the time that is spent with the shutter open, as each image must also be downloaded to the computer before the next one can be taken.

The basic process for a time lapse sequence is the same, with a few considerations. The question of how many stops between exposures becomes relevant. The disk space necessary for a sequence of images taken every few minutes over the course of an entire day can fill up the storage on a laptop computer, and capturing every other stop means that the storage requirements are also cut roughly in half. Also, if more exposures are taken to construct each single HDR frame, the time to capture the HDR frame goes up, and the amount of HDR frames captured per hour goes down, which can lead to time lapse sequences that do not sample often enough for the viewer to adequately see what is happening. For the best results, the number of stops between exposures must be balanced between image quality and speed of capture. This is probably at least partially camera dependent, since some cameras will probably be able to transfer their data more quickly than others. For example, a USB camera will not be able to compete with a Firewire capable camera in terms of data transfer, but USB cameras are much more common and affordable today.

2. HDR Image Generation

There are several options for creating a single HDR image out of multiple exposures, and they seem to yield comparable results. The most common is HDR Shop, written by Paul Debevec. This is a Windows only application. The Macintosh world has Greg Ward's Photosphere, which does essentially the same thing (and it also catalogs images). Both these applications are gui-based, so they do not handle batches well, and are therefore not well suited for time lapse photography. It's impractical to

manually construct each image, since the amount of clicking and typing would not only take hours and hours, but is a direct path to repetitive stress injury. Fortunately, Greg Ward also released hdrgen, which is a command line utility for Linux and Macintosh. This means that hdrgen and a basic perl script are the only things needed to convert the entire LDR sequence of images to a sequence of HDR images.

One consideration in generating the images is the question of the format in which the files should be written. The most common format is Radiance (.hdr), which stores its pixels in the format RGBE. One byte is dedicated to the red, green, blue, and exponent channel each. For the red, green, and blue channels, the byte describes the mantissa (the decimal part of the number), and the exponent channel is a common exponent, shared among the three color channels. This yields a tremendous dynamic range of 76 orders of magnitude, and is actually 62 orders of magnitude more than is observable in the natural world, at least here on Earth [16].

There are downsides to this format, of course. One is that precision is sacrificed for dynamic range. One byte describes the mantissa for each channel, and the exponent is shared, which means that if the exponent for two of the channels is very large, a number close to zero is not possible for the third channel. Another drawback is that negative values aren't possible, which limits the description of colors in some color spaces [16].

The newcomer format is Industrial Light and Magic's OpenEXR (.exr), which has less dynamic range than the Radiance format, but it is still sufficient to describe most of the dynamic range that can be viewed here on Earth. The format is a bit more convenient, since the read and write routines are already written in a library, and it includes "OpenEXR Viewer," which is a utility that allows the user to view different exposure segments of an OpenEXR image. This utility is not a tone-mapper, since it only displays a part of the exposure range at a time. OpenEXR seems to be

quite popular considering its newcomer status, and graphics hardware manufacturers NVidia and ATI are starting to handle this format natively [16].

OpenEXR uses its own 16 bit data type called "Half," so called because it is half the size of a 32 bit float. It achieves this savings in storage size by sacrificing precision in the higher number range. Near zero, the precision is that of a float, and the precision drops off with higher numbers. In fact, somewhere in the tens of thousands, precision drops off to the point where not every integer is represented. Still, this happens at a relatively high number, so it does not affect the image much if values get this high. Since there is no shared exponent, the problems associated with Radiance files are not present. In addition, negative numbers are possible, facilitating all color space representations. In fact, the OpenEXR wrapper is so flexible that it allows for any number of channels, and even stores in full floating point format if the user so desires [16].

3. Tone-mapping

Tone-mapping is the process of bringing a high dynamic range image back to a standard low dynamic range image so that it may be displayed on conventional hardware. Without this, there is no practical way to view the HDR image and get an idea about all the details. There are several algorithms available, and the one detailed here is Fattal's "Gradient Domain High Dynamic Range Compression."

In Fattal's method, all operations are done on the luminance, and at the end, the color is added back in, calculated based on the original color. The luminance is calculated by the simple formula of (R+G+B)/3. The luminance values are then normalized from 0 to 100, and the log of the luminance is taken (There is a very small number added to the luminance while taking the log to avoid the possibility of taking

a log of zero).

Next the phi attenuation map is calculated. The phi attenuation map is what brings the dynamic range down so that it will fit into a low dynamic range image's space. It is calculated at several resolutions, with each different resolution being half that of the previous map's resolution. Then all the resolutions are interpolated back to the original resolution and multiplied together to create the final phi attenuation map. Multiple resolutions are used because if the attenuation map is calculated once at a single resoultion, the gradients are defined by sharp lines, which would result in a "halo" effect in the final image similar to Tumblin's LCIS method. Multiple resolutions yield a slightly blurred map of the gradients when put together, and the "halo" effect is avoided.

Each individual resolution of the phi attenuation map is calculated by the following algorithm.

If $x < \alpha$,

$$\Phi = 1.$$

Otherwise,

$$\Phi = (\frac{x}{\alpha})^{\beta} \times \frac{\alpha}{x}.$$

In these equations, α and β are user defined values (.04 and .85 are good starting points respectively), and x is the magnitude of the gradient at the given point on the image. When the final map is calculated, the end result of the above algorithm is that any value of x which is small is left alone (multiplied by 1) or slightly reduced (multiplied by some number less than 1, but approaching 1). Values of x larger than α are reduced according to how much larger they are. Extremely large differences will be multiplied by numbers near zero, which brings them down to the range of the rest of the image. The value β affects the shape of the curve which defines how quickly

Figure 12. The final phi attenuation map and the resulting image. Dark areas have steep attenuation and white areas are not attenuated.

the phi map values approach zero. The final phi attenuation map is calculated by resampling the different resolution maps to the full image resolution, then multiplying the maps together. An example of the attenuation map and the resulting image are shown in Figure 12, and Figure 13 illustrates how α and β affect the calculation of Φ at a single resolution.

Once the final phi attenuation map is calculated, it is multiplied with the gradient of the luminance of the image. From this new value, the divergence is taken, and then integrated to get a new map of luminance values. A basic sharpening algorithm is run on the new luminance map. The exponent is taken, and then the luminance map is normalized from 0 to 1.

This grayscale luminance map can now have color reintroduced. Color is returned to the image with the following formula.

If $L_{in} = 0$,

$$C_{out} = 0.$$

Otherwise,

$$C_{out} = (\frac{C_{in}}{L_{in}})^s L_{out}$$

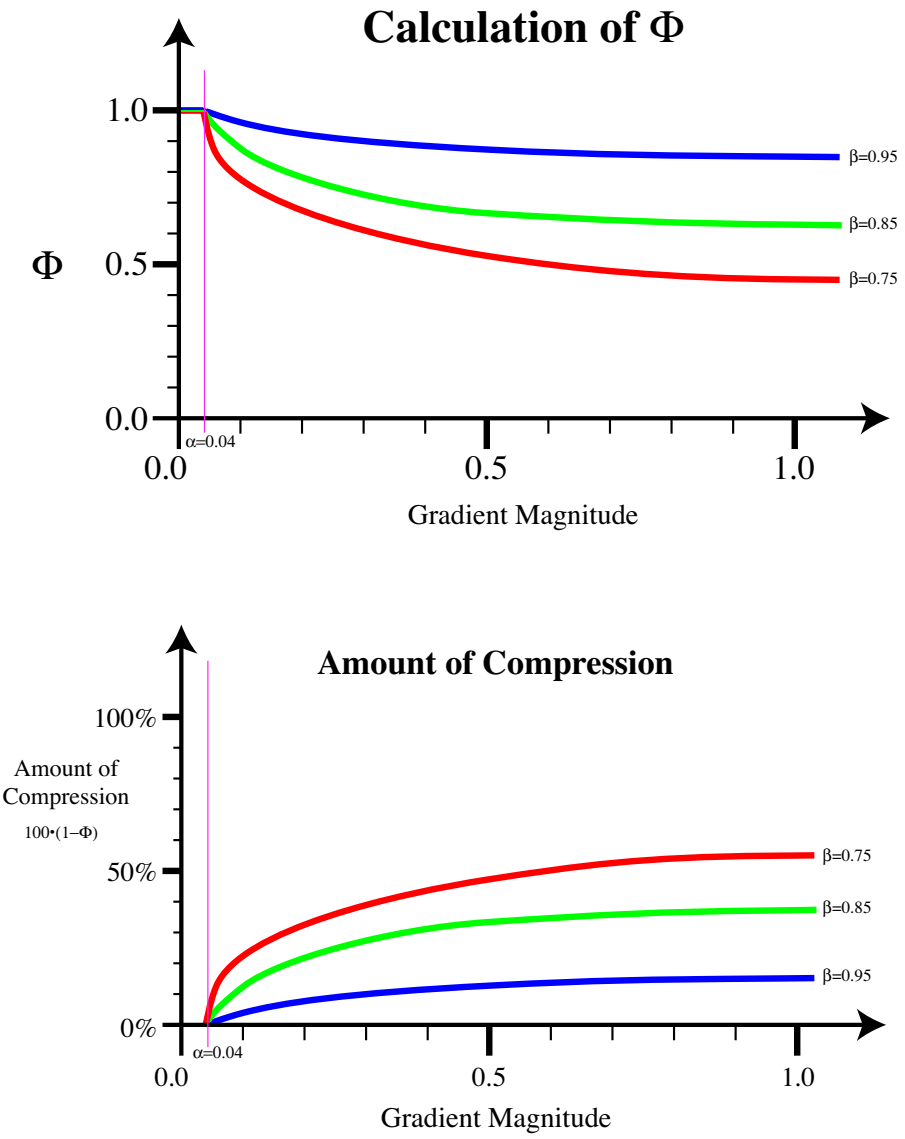


Figure 13. The "Calculation of Φ" graph (top) shows how Φ is affected by the magnitude of the gradient, and how α and β affect the Φ calculation. Note that when the magnitude of the gradient is less than α, Φ is 1. Otherwise, the shape of the curve is affected by β. The "Amount of Compression" graph (bottom) is not calculated directly while implementing Fattal's method, but it has been included because it is easier to visualize what is happening. This graph is simply 1-Φ multiplied by 100. The y-axis in this graph expresses how much a gradient will be compressed when multiplied with the phi attenuation map. Note that a gradient magnitude of less than α results in no compression.

where $C = R, G$, and B channels, and one iteration is necessary for each channel. L_{in} and L_{out} are the luminance values before and after HDR compression. The exponent s is a user-defined value for color saturation [5]. A value between 0 and 1 is required for s, and 0.5 works as a good starting point.

At this point, the image can now be stuffed back into the conventional LDR color space via old fashioned interpolation.

4. Display

For single images, display is as easy as writing out to one's favorite format and opening in one's favorite image viewing utility. Display of a sequence isn't much harder. Once the sequence has been written out with each frame in a tif or jpg format, it can be assembled into a Quicktime movie (or similar format) by several utilities or commercial software packages such as Adobe's After Effects.

CHAPTER IV

IMPLEMENTATION

This is the story of my process to assemble the tools to create a pipeline for creating sequences of HDR images and bringing them back into LDR, and then my use of this pipeline to create my own time lapse HDR image sequences.

It is the story of a journey. This journey took me from a place with no HDR images to a place with HDR images, but no way to display them. This journey then took me to a place where I could display them, but not very well, and then to a place where I could display them pretty well. But still this new place had no power to display a sequence of HDR images, but that point was irrelevant, since I had no sequence of HDR images to display, anyway. And then, as if fate was my guide, this journey came to a place where I had both a means to display a sequence *and* a sequence to display.

This journey was fraught with intrigue, friendship (new and old), beauty, a bit of cliche, and not much danger. In fact, the only real danger is the danger of beating this metaphor into the ground, and hence I shall get on with my story post haste.

1. Getting Started

It all started innocently enough, as I had shown my advisor some paintings by Donald K. Lake that I admired, and upon which I wanted to base a different project. These amazing paintings capture the scene in a steel mill better than a photograph could, because of the amount of detail captured from shadows up through brightly lit areas and even into flame. Two of his paintings are shown in Figure 14 and Figure 15. My advisor, too, was impressed by them, and said that their photographic look reminded

Figure 14. *Industry: Chamber*, a painting by Donald K. Lake that has a similar look to HDR tone-mapping. Reprinted with artist's permission.

him of some photographic images that looked almost like paintings. He showed me the SIGGRAPH 2002 book, which contained some striking images that were both photographically realistic, and hyperbolic – a bit surreal.

These paintings and the images in the SIGGRAPH 2002 book had a few things in common. First, they were both very realistic in terms of the forms and textures on the objects in the scene. And second, they were both surrealistic in the way that they treated light. Both captured details in dark places, along with details in bright areas in such a way that a photograph never could. They represented a range of light that simply wasn't possible in photographs. Since Morpheus did not appear to offer me a red pill or a blue pill, I knew that I would need to implement one of these SIGGRAPH papers in order to enter this strange new world.

Tone-mapping was new to me as an application for HDR images. In the past,

Figure 15. *Industry: Blast Furnace*, a painting by Donald K. Lake that has a similar look to HDR tone-mapping. Reprinted with artist's permission.

I had only heard of the use of HDR images as tools for capturing the illumination or environment mapping of a scene for 3D computer generated images, often called light probes [3]. I knew the first step was to create a single HDR image, and I knew the college wasn't about to cough up the coin for hardware to capture HDR images in one shot, so I set about creating my images via the multiple exposure method.

I set up a Canon Powershot S200 on a tripod and took multiple exposures of the mess that is called my work area. It included a lit lamp, my computer monitor, and various and sundry items littering the desk. I sat down to write a program to analyze the images and add up the intensities to create a high dynamic range image. This program, however, did not reach fruition, though it did get me started on my journey.

2. Formats

One obstacle to writing this HDR creation program was the format in which to store the data. This was easily overcome with the help of OpenEXR, the open-source HDR format created by Industrial Light and Magic [6]. OpenEXR provides a library for read and write operations, and also offers the option of reducing the amount of storage required by using the "half" data type. Half is a floating point data type, but it cleverly rearranges the way data is stored so that it only uses 2 bytes for a number, as opposed to 4 bytes for a float. It does this by sacrificing the precision of numbers with higher values. More information is available at http://www.OpenEXR.net. OpenEXR also has a viewer that allows the user to view a clipped LDR version of the image. The user can select what exposure to view, and thus confirm that an OpenEXR file contains the data that it should.

The other major obstacle I encountered was the fact that most digital cameras

do not store images linearly. A linear representation means that doubling the light intensity should double the pixel value. Instead, the cameras have been tuned to look more pleasing to the human eye, which means that in order to determine how much light should be represented by a particular pixel, the camera's response curve must first be taken into account[3]. This involved writing code to solve a very large matrix or using Debevec's Matlab code [3]. Not being familiar with Matlab, I decided that it was not my place to reinvent the wheel, and in the name of expediency, used Paul Debevec's HDR Shop to put together my image.

HDR Shop had a few drawbacks, however. One was that it was not geared toward batches of frames. To make a single HDR image in HDR Shop required a lot of mousing and clicking. Something more efficient was needed. Another drawback to HDR Shop is that it is a Windows only application. Since this is not my platform of choice, and since it crashed frequently while assembling images (a feature I attribute wrongly or rightly to the operating system), I needed a Linux and/or Macintosh solution. Fortunately, I found it with Greg Ward's hdrgen, which is available for both Linux and Macintosh. The fact that it is a command line application also meant that I could set up a script to do batches of conversions when the time came (It is worth noting that about the time I finished putting together my pipeline, a new version of HDRShop was released which handles batches of frames. However, this version is no longer free, even for educational use).

Armed with my newly acquired tools, I set about my house taking picture sets with both dark areas and bright areas. I assembled them with hdrgen and viewed them with OpenEXR's viewer. This utility is not a tone-mapper, and does not let the user view an entire range all at once. The user is able to pick a particular exposure contained in the .exr file, and view the dark area, for example. The midrange area can then be viewed, and then the bright area, but only one of these ranges at a time.

This brought me joy. But as amusing as this was, something was missing. I wanted more than just one range of exposure at a time – I wanted it *all,* and I wanted it in *one exposure.* I began to look longingly at the SIGGRAPH 2002 book lying on the shelf. *It was time.*

3. Implementing Fattal's Method

If I looked hard enough (and I did), it was possible for me to find all the pieces to my pipeline puzzle on some platform or another, in various stages of user friendliness and completeness. The exception to this is the tone-mapping software. I looked around, asked various people who had written papers on the subject if they would share their software with me, and came up empty every time. The best response I got was, "Sorry, it's not ready for public release – maybe later," but the average response was to be ignored completely. However, since this is the key software element to my project, it was fitting that I create my own version, and add functionality to do batch processing. Fattal's method for tone-mapping was my choice because it seemed to give the best results, and the algorithm appealed to me because it seemed clever and logical.

At this point, I will take the literary device of foreshadowing to an extreme and divulge that I got something working. Hopefully this comes as no surprise to the reader, but perhaps the reader saw some of my early code and therefore had some doubts. I got a crude and error prone version working fairly quickly. There was plenty of noise and the sunlit areas were a bit blown out, but it did capture details in dark areas through moderately bright areas. I sent a picture of my kitchen to my advisor who was at a conference overseas. Despite my disappointment that my results didn't look like the results in the SIGGRAPH paper, my advisor seemed overjoyed at the

results, and he hadn't even seen my early code.

I wrote to Fattal, and was quite surprised when he not only wrote back, but offered some advice. There were several noteworthy items. First of all, he cleared up some unresolved questions I had about the phi calculation equation that was listed in the SIGGRAPH paper. The way it was written, it was possible for a calculation to result in division by zero. This is, to put it in complicated mathematical mumbo jumbo, "bad." Or to put it in layman's terms, division by zero yields unpredictable results. I got around this by overriding my phi calculation and setting phi to 1 in such a case, and Fattal confirmed that this is what he and his group did. In fact, Fattal had updated his method to include setting phi to 1 any time the magnitude of the gradient is less than α, which renders the division by zero question moot.

Fattal also divulged a new trick since the release of the SIGGRAPH paper: Normalization of the image to 100 before working on it. This is actually quite handy (if not necessary) because the value for α suggested in the SIGGRAPH paper is designed to be a noise threshold. If the paper's suggested value of 0.1 is used on an image which ranges from 0.0 to 1000.0, α is set to 0.01% of the range of values. If the same image is normalized from 0.0 to 1.0, an α of 0.1 covers 10% of the range of values. This obviously yields very different results. Normalization to 100 means α always affects a similar range of values.

And speaking of that recommended α value, the SIGGRAPH paper suggests that a value of 0.1 is a good starting place, but Fattal suggested a value of 0.01 with the new technique of normalizing to 100. This works much better, and further tweaking on my part showed that a better starting α for the way I had things set up was actually 0.04. I arrived at this value, as well as the β value of 0.85 by starting in Fattal's recommended range and experimenting with different images and different values.

Still, while these things helped a bit, I hadn't yet achieved results that came close to Fattal's. And of course, Fattal's answering my questions only furthered that kind of question asking behavior. I wrote back with more, and he shared some of his code with me to confirm that I was doing everything the way he was. In general our code did the same thing, though his was usually more elegant. I "open-sourced" a passage of his code that was more elegant than mine – the divergence calculation. "Open-sourced" in this context is a euphemism for "borrowed," which in turn is a euphemism for "copied, while giving credit."

I still was not satisfied with the results, though, and SIGGRAPH was approaching. I wrote to Fattal again, and he agreed to meet with me after his 2004 SIGGRAPH presentation. An hour before our meeting, I was going over my code, removing obsolete code and commenting things to make it easier for him to help. Then I found a major problem. It was necessary to view the image at various stages to check that things were going as planned. For this, the exponent of the image was taken (undoing the log calculation) in order to compare the original with the version in progress. Unfortunately, the fact that the log of the image was no longer being manipulated was forgotten, so things were not set back to where they should have been before continuing with the calculations. Integrating followed by taking the exponent of the image (again) muddied up the image. The person responsible for this was severely reprimanded, but unfortunately couldn't be fired because of connections he or she has with the project coordinator.

This discovery put me within reaching distance of my goal, but I was still not there. At last I had something that looked comparable to Fattal's image; I was 90% there. I hoped that whatever advice he could offer would get me the 10% boost I needed. He went over my methods and my code, and worked with me for at least an hour, but in the end no solution was found. I returned home from SIGGRAPH

90% complete. I stagnated there for a while, before deciding to push on with the user interface for multiple frames.

I wanted the user interface to work on Macintosh, my platform of choice, but I also wanted it to work cross-platform, so I decided to use gtk, a gnome-based interface. The graphical user interface was my first, so I went through several iterations, and spent almost as much time learning gtk and working on the user interface as I did the tone-mapping program. Eventually, I reached a point where I could load sequences of images, set keyframes for my values, and apply them to the sequence. My software, though error ridden, unstable, and imperfect, was "functional." I dubbed it "gct (Gradient Compression Tool)" and left it in lower case because it made me feel like a real linux programmer.

4. Image Capture

I set out for something to run through my software to test it in earnest. I began to plan out sequences to capture. It is worth noting here that capturing any significant quantity of images manually is next to impossible, as changing the shutter speed for every exposure is awkward at best. Assuming it's possible to change the shutter speed hundreds, or possibly thousands of times for a sequence in HDR without losing one's sanity, it is highly unlikely that it's possible without bumping or jostling the camera slightly, which would show up when the frames are played back. It is also important to get the different images that are used to construct a single HDR frame as quickly as possible. This minimizes changes in the scene that occur between frames, such as candles burning down, the sun (and therefore shadows) shifting slightly, and objects being disturbed by other things. It also allows more HDR frames to be taken per hour if the time to capture a single HDR frame is reduced, which means smoother

motion in the playback for many subjects.

The only practical solution is to hook up the camera to a computer via USB (or FireWire) cable and control the camera remotely. This also addresses problems with memory cards, since the memory card in cameras is insufficient to capture enough exposures to do a time lapse of any significant length in HDR, so it is necessary to download images as they are captured. This also eliminates the problem of bumping or moving the camera slightly when adjusting the shutter speed. But even when hooked up to a computer, adjusting the shutter speed via mouse and keyboard is too repetitive a task to do be practical. The possibility of moving the camera is gone, but the possibility of clicking the wrong speed, naming the file improperly, or overwriting other files is too great. The capture process simply must be automated.

After creating one of my first single HDR images by manually adjusting the shutter speed to capture 13 LDR images, this was obvious to me. I came to the conclusion that it was inevitable that I would have to write software to control the camera, so I sent away to Canon and Nikon for their SDK (Software Developer Kit), which allows users to write programs to control the camera by a computer. When it came time to begin coding, I started looking for advice on Canon's SDK, since that was the brand most readily available to me. Suddenly, Google turned up a new piece of software by Sean O'Malley at the University of Houston called AHDRIA (Automatic High Dynamic Range Image Acquisition), which controls Canon cameras and takes a sequence of LDR images specifically designed for HDR image creation [9]. AHDRIA controls the aperture, shutter speed, interval between exposures, and captures all the frames as fast as the USB interface will allow.

The catch, of course, is that it is a Windows only application. Suddenly I was faced with one of the hardest dilemmas of my life: *Should I spend weeks coding to reinvent the wheel, or go to the Dark Side and use a Windows application, no matter*

how free of charge? The clock, ticking like the Telltale Heart, told me to sell out faster than a broke and desperate Benedict Arnold.

I procured a Windows laptop from the college and began capturing data for HDR image sequences. Not having been designed for image sequences, Mr. O'Malley's software lacked the full spectrum of functionality that I required. It only captures a set of LDR images for one HDR image at a time. Once the program is set up, however, it only requires only one click to capture the data for the next frame, so it is certainly usable for capturing a time lapse sequence. One must only sit and click the mouse at whatever interval the user decides is appropriate.

After a few image sequences clicking every five or ten minutes, I realized that there are more interesting ways to spend an afternoon and evening, not to mention that the resulting movie was often a bit jumpy because the button was sometimes clicked a minute or so late. A *completely* automated version of AHDRIA was needed, and I lobbied Sean O'Malley heavily to add this feature. Though very helpful and friendly, he understandably possessed neither the time nor the desire to tweak his software at my bidding, as my new friend had his own education to finish.

This is where my old friend Robert James Knopf stepped in (Actually, he also stepped in earlier when he recommended gtk as a tool for writing a user interface). He is familiar with Windows programming and wrote a utility to simulate a click on the AHDRIA button at whatever time interval I specified. I declared that my image capture pipeline was now complete, and nothing would stop it from working seamlessly.

Naturally, after a declaration like that, the pipeline had no choice but to fail. The problem seems to lie in the communication between Canon's cameras and the software. The SDK seems to work better with some cameras than others. Since he is a student, and developed AHDRIA on his own time and money, Sean O'Malley could

not afford to test his software on every camera in Canon's line, so cameras are tested by the first user to test a particular model (a.k.a. me).

a. Image Capture – A Line of Cameras

I started with the Canon S200. AHDRIA recognizes this camera and works with it, but there is a fatal flaw: After capturing roughly five sets of images, the camera locks up. It stops downloading data and responding to commands. The only way around this is to shut off the camera between each set of exposures and restart it. This has two drawbacks which make it unusable for time lapse HDR photography: First, it makes full automation impossible, since the user must be there to turn the camera off and on each time, and the camera must be reconnected to the software every time it comes back online. Second, the S200 has a lens that retracts into its body every time it is shut off. When it is turned on, the lens comes out, but not *precisely* to the same point every time. The result is a movie that has a slight jumpiness to it, as all the frames don't line up perfectly.

This did not appear to be a major setback, since the S200 is a lower level camera, and isn't even available anymore. I intended the S200 as a test camera only, and planned to do most of my work with a Digital Rebel or 10D, both of which are digital SLR cameras. I was looking forward to experimenting with different lenses on the SLR cameras.

Unfortunately, the Digital Rebel and 10D did not function with AHDRIA. In fact, while they functioned with "Remote Capture," Canon's own software for controlling a camera from a computer (single images only – not suitable for HDR capture), there were some interface problems with the 10D. The LCD display on the 10D did not appear to work when hooked up to a computer. The end result of all this is that the

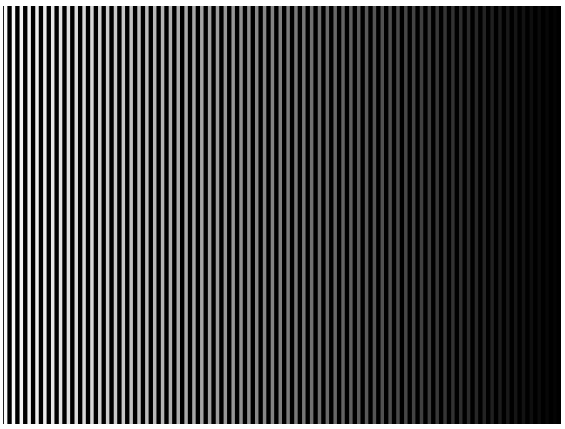
Digital Rebel and 10D were unusable with AHDRIA, even though Sean O'Malley was kind enough to try and update his software to work better with Canon's drivers.

Though the S410 is similar to the S200, I tried that next, since it would allow me to experiment with higher resolutions. Unfortunately, this camera did not work either. Eventually, I was able to borrow a Canon G3, which worked quite well. The lens does retract partially into the body when the camera is turned off, but this was not a problem, as the camera does not freeze up with repeated captures, and therefore the camera is not turned off between exposures. The G3 became my workhorse camera.

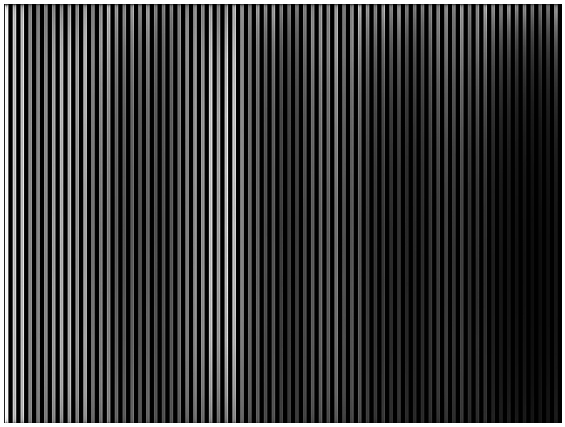
5. Gradient Compression Tool Revisited

Unfortunately, while pursuing other parts of the project like image capture and subject matter, the bugs in my software did not fix themselves. This meant I needed to figure out the problem on my own (with the help of my advisor and colleagues). My advisor suggested an "artificial" image – one that was not made from photographs, where I knew the values going in, and should be able to predict the values going out. This was a wonderful idea, so I created some vertical stripes on a black background. The stripes were a constant brightness along the stripe itself, but each stripe was a different brightness, ranging in intensity from a value of 1.0 to 71.0 (arbitrary floating point values I chose). A representation of the image is in Figure 16 A.

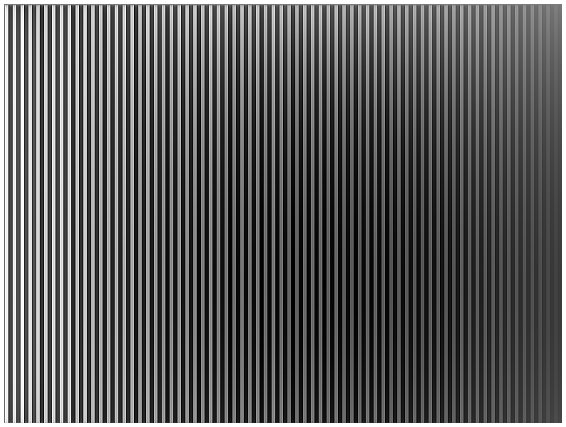
If all was working flawlessly, the image should have been a black background with all the stripes being roughly the same brightness after running through gct, since the only gradient changes in the image are along the borders of the stripes, and each stripe would be compressed independently of the others because they were not touching. This is essentially the result I got, with one notable exception: The image had dropouts in it, particularly at the borders. This was not the type of problem I



A



B



C

Figure 16. A: A representation of the test pattern used to debug. The background is black, with the stripes ranging in value from 1.0 to over 70.0. These values are floating point values, not integers. B: Results with divergence calculated wrong. There should be no dropouts if things are working properly, and ideally, the lines should be a consistent brightness from top to bottom. C: Results with proper divergence calculation. The dropouts are almost gone, and lines are more consistent vertically.

expected from this algorithm, since these dropouts did not appear to coincide with the details of the image. I expected problems with the algorithm to yield results that affected each stripe in its entirety, but these dropouts appeared only on some bars, and mostly in the center of the bars. It looked as if the image was fading along the center of the stripes, particularly at the borders (Figure 16 B).

Since this didn't make sense to me, I decided to try and isolate where the problem was taking place. Was it the compression or the integration? The phi attenuation map showed what I would have expected for a working algorithm – a change of compression along each stripe, with the same values running the entire length of the stripe. If the values of the phi compression map were not constant from top to bottom, this would explain the dropout, but each and every stripe had a constant attenuation from top to bottom. This means the problem was in the integration stage.

This narrowed the problem down to either the integration function, or how the data was being fed into the integration function. Unfortunately, creating my own two dimensional integration function was beyond my ability (at least if I wanted to graduate before my seven allotted years), so I used the function published in Numerical Recipes [10]. I read the chapter, and gained a general understanding of how it works, but was still in no position to improve upon their code. Thus the integration function is somewhat of a "black box," meaning I can't significantly modify how it works.

Fortunately, I found a major problem with the way I was feeding data into it. The problem was in the elegant code from Fattal that I "open sourced." While it was more elegant than mine, I didn't notice that he had been experimenting on it, and he commented out a line where he divided by a number. I didn't understand at the time that this was necessary for the proper calculation of the divergence. Upon discovering this, my results improved dramatically as shown in Figure 16 C. It's amazing what

happens when proper calculations are used.

Finally, my tone-mapping results were comparable to Fattal's, I had a working automated method for collecting data, and my software was ready to handle image sequences. The images do often benefit from a histogram adjustment in Photoshop, as the histogram adjustment in gct is primitive (and is a topic for future work). This process, though not strictly necessary, can be automated, as well. My pipeline was now complete, and so it was time to capture some good subject matter.

6. Subject Matter

To simply set up something like a flower in direct sunlight and make a time lapse sequence while it opens would not be taking full advantage of this pipeline. Time lapse HDR photography is interesting in scenes that contain both dark areas with detail and bright areas with detail, or scenes with dramatically uneven lighting. I tried to choose subject matter that would take advantage of these.

At the same time, there are limitations to this process. It takes between 35 seconds and 4 minutes to capture a single HDR frame's worth of data with the camera I had available to me. The variation in the capture time depends upon how many exposures are taken, which affects the quality of the final image. The fact that it takes so long to capture data for one HDR frame means that subject matter is limited. Moving objects need to move slowly enough that no significant change takes place during the capture of one HDR frame's data, or artifacts will be introduced. The artifacts may be desirable, as in my HDR time lapse sequence in a restaurant, where I wanted the patrons to be visible only as blurs. But in general, such artifacts are usually not desirable, and will show up whenever objects move between the LDR exposures that make up a single HDR frame. For this reason, people and animals

usually don't work well. It is extremely difficult for even a willing subject to sit still enough that no artifacts are introduced.

It is also important to pick an event that happens slowly enough that HDR frames can be created quickly enough to make smooth motion in the image sequence playback. I encountered one such problem in an image sequence where shadows were being cast by branches and leaves blowing in the wind. The shadows were moving around so quickly that they just appear to jump around randomly in the final image sequence.

CHAPTER V

RESULTS

The results of Fattal's method, and therefore gct, yield a look that is somewhere between photography, painting, and computer generated images. This is because it is an image with the detail of a photograph, but is describing scenes that the viewer has never been able to see in a photograph. The person viewing knows that there is something unusual about the image, but can't immediately decide what it is. Invariably, while I was working on final images, if a person who was unfamiliar with the work walked by, they would look at the image and ask if it was a computer generated image. Personally, I find the results to look somewhat idealized, like a Norman Rockwell painting.

Some things seem to lend themselves well to the process. I found that a romantic dinner scene, lit by candles, christmas lights, and lights on artwork looked very interesting (Figure 17). While it was too dark for someone sitting in the actual scene to comfortably read, the darkness of the scene is ambiguous, even mysterious in the final image sequence. The user is definitely not under the impression that the scene is brightly lit, but the image does not seem particularly dark, either, since detail is visible everywhere, even in corners and under the table. Yet, the image maintains the characteristics of a dark place. For example, small lights and candles can cast a glow on objects around them, which wouldn't be noticeable in a brightly lit room, and colors seem subdued as they are in lower light situations. I wanted to capture the candles burning down, a subject which works surprisingly well when the air is still around the candles, so the flame doesn't jump too much, and results in a smooth capture.

Figure 17. Single image in a time lapse series captured by the Canon G3, assembled with hdrgen, and tone-mapped in gct.

Another interesting scene I wanted to capture was something in front of a window, where in conventional time lapse photography, the subject or the background would not be exposed properly. For this, I set up some flowers and trinkets on a window sill. The flowers open up over the course of the image sequence, which takes place over the course of an afternoon. The sun also moves from overhead to the horizon, and while the sun itself is not in the sequence, the shadows of buildings and trees around tell the story of the passage of time. Detail is visible both in the outside world, and in the objects on the window sill, which normally would not be properly exposed with the outdoor part of the scene (Figure 18).

I wanted to try a more chaotic scene than something I set up completely artificially, so I asked permission from a local restaurant to come in and take some pictures. The Blue Baker, which not only turns out to be a very friendly place with

Figure 18. Six frames from the "Flowers" time lapse sequence. The flowers open and the shadows move as time passes.

Figure 19. Six frames from the "Blue Baker" time lapse sequence. Note that the patrons are blurred if they are moving when different exposures for the HDR frames are taken.

tasty sandwiches, also has large windows suitable for a time lapse HDR experiment. I captured an image sequence in front of one of these large windows. Everything including tables in the interior of the building, a light bulb on the interior, tables on the patio, the parking lot, and buildings in the distance are visible. I set up mid morning and captured the shadows from the sun moving across the empty parking lot. Then, around noon, the scene explodes with action, including the parking lot filling up, people milling around, and sitting down at tables to eat. Every person in the scene is a blur, as they are constantly in motion. The only exception is me, sitting very still right in front of the camera for the last few frames (Figure 19).

I had poor results capturing a moonrise, not because the subject matter wasn't good, but because of the limitations of the camera and the way the AHDRIA software worked with it. The camera tried to refocus between sets of frames, and it couldn't find the proper focus for the moon. A camera with a manual override for focus would probably work for this, and is something I'd like to try again in the future with image capture software improvements and a more sophisticated camera.

CHAPTER VI

CONCLUSION AND FUTURE WORK

Though there is room for improvement, this approach for creation of time lapse HDR image sequences works quite well. Once the tools are assembled, the pipeline makes the process efficient, and minimal user input is required to create the image sequences. This frees the user to concentrate more on subject matter, rather than the tools for creating the images.

1. Future Work

There are many ways to improve this process, including hardware, software, and exploring more subject matter. For hardware improvements, I imagine a camera capable of capturing enough data for high dynamic range images in a few seconds. It could be a commercially built camera, or one constructed from several conventional cameras firing at the same time. Perhaps the parallax between cameras could be overcome by mirrors.

A simpler solution, of course, would be to find a camera with higher download speeds. Since the majority of the time capturing is spent downloading, not exposing the camera's CCD, perhaps a FireWire camera could download almost as fast as the image is captured, meaning a much higher frame rate would be possible.

Given more time, I would like to either create my own version or port Sean O'Malley's AHDRIA to the Macintosh platform, and add features specific to time lapse capture, such as complete automation for multiple HDR images, and perhaps even immediate generation of the HDR image, as opposed to doing it later via perl script and hdrgen.

But the most pressing software update, would be improving the user interface of gct and increasing functionality. I would like to add separate saturation controls for each color channel, rather than just having one global channel. Also, the histogram adjustment is currently crude. Controls for setting the histogram midpoint would help quite a bit.

Continuing the list of gct user interface complaints, the batch processing user interface is awkward and needs to be reworked to be more user friendly and offer more control. Keyframes should be visible in a timeline, similar to the one in Maya. Immediate feedback of what an image would look like in a small section of the image would be very helpful, rather than having to process the entire image in order to see results. Thumbnails of the image would be helpful. The code needs to be optimized to run more quickly, and further investigation of the integration method might improve the final image. And finally, a progress bar with an option to cancel the process would also reduce user frustration.

Future HDR image sequence subject matter is very enticing. An overhead shot of a mall at Christmas shopping season, particularly a place where people will be standing in line and moving only occasionally would be fun. An indoor location with skylights, picture windows, and lots of Christmas lights would be a great place to start. Perhaps the flow of people at an airport, with the flow of planes right outside the window, or sunrise through sunset in rugged terrain. With the proper camera, it would be possible to capture the moonrise over some prominent buildings.

2. Other Applications

Another application for this technology (or parts of it) is the possibility of using gct for HDR environment map previews. Environment maps are often captured in

high dynamic range to use for lighting a scene. At SIGGRAPH 2004, for example, Paul Debevec et al. presented an animation featuring The Parthenon, lit with HDR environment maps. The maps were collected as time lapse HDR photography from images of the Los Angeles sky [12], but differed from this research in that they were meant to be used as environment maps, not viewed directly. But if a user collects a time lapse sequence of environment maps and wants to visualize what's going on, gct can be useful in letting the user see where light sources are, and what's going on in the environment. This will allow the user to ensure that no surprises or mystery light artifacts will occur at render time, because the animated environment maps can be checked for content visually.

REFERENCES

[1] K. Nordine. (1995). White. On *Colors.* [CD]. Asphodel.

[2] Leonardo Da Vinci. *The Dreyfus Madonna.* Scanned by Mark Harden at: http://www.artchive.com. Accessed December 2004.

[3] P. Debevec and J. Malik. "Recovering High Dynamic Range Radiance Maps from Photographs." *SIGGRAPH 1997 Conference Proceedings on Computer Graphics,* pp. 369-378, 1997.

[4] F. Durand and J. Dorsey. "Fast Bilateral Filtering for the Display of High-Dynamic-Range Images." *Proceedings of ACM SIGGRAPH 2002*, pp. 257-266, 2002.

[5] R. Fattal, D. Lischinski, and M. Werman. "Gradient Domain High Dynamic Range Compression." *Proceedings of ACM SIGGRAPH 2002*, pp. 249-256, 2002.

[6] Industrial Light and Magic. OpenEXR. [Software Library]. Available at: http://www.openexr.net. Accessed 2004.

[7] Donald K. Lake. *Industry: Blast Furnace.* [Scanned Slide]. Champaign, IL. 2004.

[8] Donald K. Lake. *Industry: Chamber.* [Scanned Slide], Champaign, IL. 2004

[9] S. O'Malley. AHDRIA. [Software]. Available at: http://www2.cs.uh.edu/~somalley/hdri_images.html. Accessed 2004.

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C++*, pp. 829-890, New York: Cambridge University Press, 2002.

[11] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. "Photographic Tone Reproduction for Digital Images." *Proceedings of ACM SIGGRAPH 2002*, pp. 267-276, 2002.

[12] J. Stumpfel, A. Jones, A. Wenger, C. Tchou, T. Hawkins, and P. Debevec. "Direct HDR Capture of the Sun and Sky." Available at: http://www.ict.usc.edu/graphics/skyprobes/. Accessed November 2004.

[13] J. Tumblin, and H. E. Rushmeier. "Tone Reproduction for Realistic Images." *IEEE Computer Graphics and Applications,* vol. 13, no. 6, pp. 42-48, November 1993.

[14] J. Tumblin, and G. Turk. "LCIS: A Boundary Hierarchy for Detail-Preserving Contrast Reduction." *Proceedings of ACM SIGGRAPH 1999*, pp. 83-90, 1999.

[15] J. Tumblin, J. Hodgkins, and B. Guenter. "Two Methods for Display of High Contrast Images." *ACM Transactions on Graphics*, vol. 18, no. 1, pp. 56-94, January 1999.

[16] G. Ward. "High Dynamic Range Image Encodings." Available at: http://www.anyhere.com/gward/hdrenc/hdr_encodings.html. Accessed November 2004.

[17] G. Ward Larson, H. Rushmeier, and C. Piatko, C. "A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes." *Visualization and Computer Graphics*, vol. 3, no. 4, pp. 291-306, 1997.

APPENDIX 1

Several image sequences discussed in the thesis can be viewed in the "movies" file.

VITA

Brian Clark
2N962 Grand Monde Dr.
Elburn, IL 60119
bc@shout.net

Education

M.S. in Visualization Sciences	Texas A&M University, May 2005
B.S. in Media Studies	University of Illinois, May 1992
A.S. in Computer Visualization	Parkland Community College, 2000